Challenge Lesson: Ozobot Evo—Decimal to Binary Converter

By Richard Born
Associate Professor Emeritus
Northern Illinois University
rborn2@niu.edu

Students of computer science need to become adept at algorithm design and the use of user-defined functions. They also need to understand how to convert decimal numbers to their equivalent in the binary number system—the ultimate language of digital computers. This challenge lesson will provide the student with lots of practice in these skills.

There are a variety of techniques for converting a decimal number to binary. One of the most common is successive divisions by 2, each time noting the quotient and remainder. Figure 1 illustrates this procedure starting with the decimal number 115.

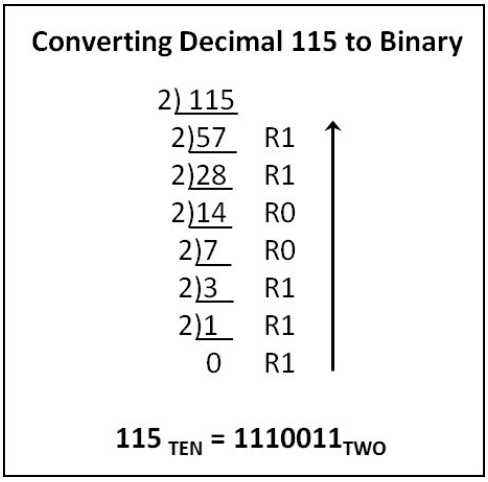

Figure 1

The procedure begins by writing the decimal number in an upside-down division symbol, and writing 2 as the divisor to the left of the symbol. When 115 is divided by 2 the quotient is 57 and the remainder is 1, indicated by R1 to the right of 57. 57 is then divided by 2, giving a quotient of 28 and a remainder of 1. This process continues until the quotient is zero, where we see that 1 divided by 2 gives a quotient of zero, and a remainder of 1. As shown at the bottom of Figure 1, the remainders are then written next to each other, starting with the remainder at the bottom of the list of remainders and ending with the remainder at the top of the list.

Figure 2 shows that 1110011_{TWO} can be obtained by adding the base 10 numbers 64, 32, 16, 2, and 1, for a result of 115_{TEN}.

1	1	1	0	0	1	1
64	32	16	8	4	2	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 2

There are two OzoBlockly blocks that are particularly helpful when considering how to implement the "division by 2" algorithm. They are shown in Figure 3.

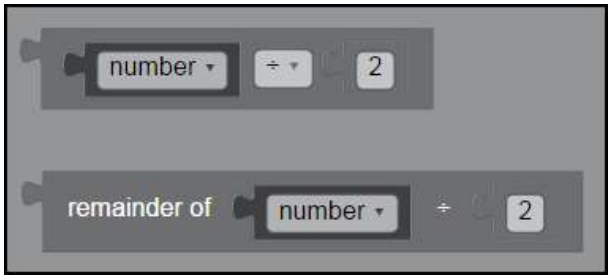

Figure 3

You need to remember that OzoBlockly division is integer division, and as such the division process results in the quotient with the remainder missing. Thus 115/2 = 57 in OzoBlockly arithmetic, not 57.5. If you want the remainder when dividing a number by 2, OzoBlockly provides the "remainder of" block, as shown at the bottom of Figure 3. The remainder when 115 is divided by 2 will be 1.

The Map for this Challenge Lesson

The real challenge when designing your OzoBlockly program is to come up with a way by which Ozobot Evo can display the corresponding binary number in some way while moving about the map. Figure 4 shows a small copy of the map that we will use to discuss how your program should cause Evo to move around on the map. A larger copy, suitable for printing and using with Evo and your OzoBlockly program, can be found on the last page of this document.

You will immediately notice that the map consists of two main parts. In the red portion at the top, Evo starts at the location labeled "Start" and facing the direction of the gray arrow. Evo then stops at the first intersection and speaks a random three-digit decimal number between 1 and 127. Evo then moves on to the second intersection where he waits for the user to press

Evo's button. This gives time for the user to determine (on a piece of scratch paper, if necessary) the binary equivalent of the random number. Evo then proceeds to the blue portion of the map stopping at blue intersections and speaking "zero" or "one" as appropriate for the given random number. He does not stop at intersections with leading zeros. For example, for 0001101, Evo will NOT stop at the intersections for 64, 32, and 16, and will not speak "zero" when at these intersections. As soon as Evo reaches the most significant 1 (the leftmost 1 in 0001101, he stops and speaks "one" and then speaks the remaining digits of the binary number as he reaches each of the remaining intersections.

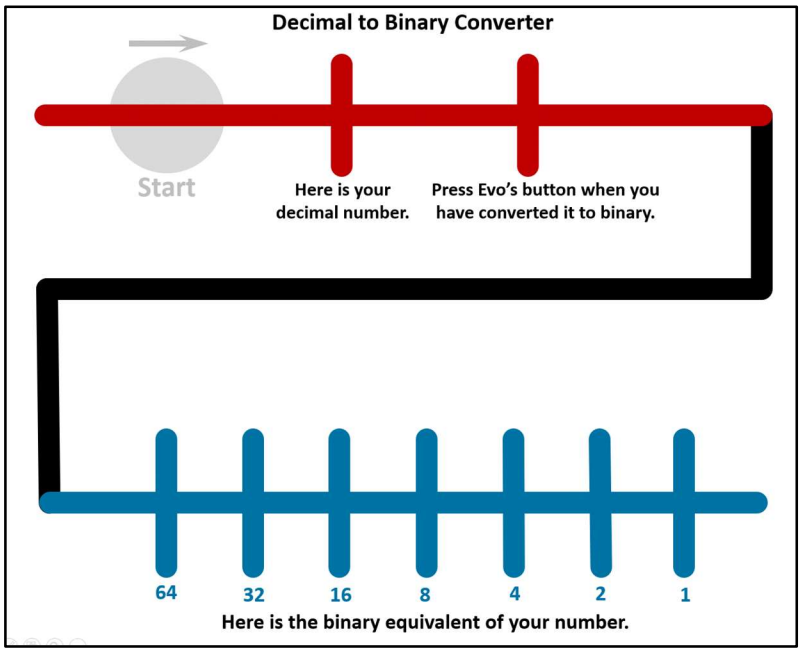

Figure 4

Your Challenge

This challenge is complex enough that the use of functions (also known as procedures) will help to break the coding into more manageable parts.

Use the "successive division by 2" algorithm discussed earlier to create a function that will convert *the random number* to its binary digits 0 or 1. These binary digits must be stored in an array of length 7 (with array indices 0 through 6). Figure 5 shows what a call to this function might look like.

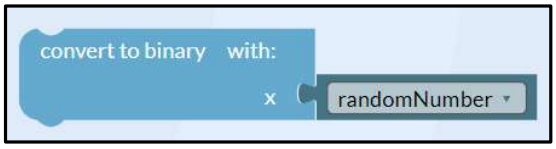

Figure 5

Decimal to Binary Converter